
Preface

This book has four major themes: *algorithms*, *data structures*, *abstract data types (ADTs)*, and *object-oriented methods*. All of these themes are of central importance in computer science and software engineering.

The book focuses on a number of important ADTs – stacks, queues, lists (sequences), sets, maps (tables), priority queues, trees, and graphs – that turn up again and again in software engineering. It uses these ADTs to motivate the data structures required to implement them – arrays, linked lists, search trees, hash tables, and heaps – and algorithms associated with these data structures – insertion, deletion, searching, merging, sorting, and traversal. The book shows how to implement these data structures and algorithms in the object-oriented programming language Java.

The book's typical approach is to introduce an ADT, illustrate its use by one or two example applications, develop a 'contract' specifying the ADT's values and operations, and finally consider which data structures are suitable for implementing the ADT. This approach clearly distinguishes between applications and implementations of the ADT, thus reinforcing a key software engineering principle: separation of concerns. The book motivates the study of data structures and algorithms by introducing them on a need-to-know basis. In other words, it adopts a practical (software engineering) approach to the subject, rather than a theoretical one. It does not neglect the important topic of algorithm analysis, but emphasizes its practical importance rather than the underlying mathematics.

Object-oriented methods have emerged as the dominant software technology during the last decade, and Java has become the dominant programming language in computer science education. These developments call for a fresh approach to the teaching of programming and software engineering. This book takes such a fresh approach, using object-oriented methods from the outset to design and implement ADTs. It avoids the mistake of reworking an older book on algorithms and data structures that was written originally for Pascal or C.

Several of the ADTs introduced in this book are directly supported by the Java 2 collection classes. Programmers will naturally prefer to use these collection classes rather than design and implement their own. Nevertheless, choosing the right collection classes for each application requires an understanding of the properties of different ADTs and